



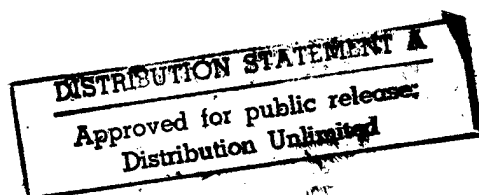
Carnegie Mellon University
Software Engineering Institute

Workshop on COTS-Based Systems

Patricia A. Oberndorf
Lisa Brownsword
Ed Morris
Carol Sledge
(Editors)
November 1997

SR

Special Report
CMU/SEI-97-SR-019



Carnegie Mellon University does not discriminate and Carnegie Mellon University is required not to discriminate in admission, employment, or administration of its programs or activities on the basis of race, color, national origin, sex or handicap in violation of Title VI of the Civil Rights Act of 1964, Title IX of the Educational Amendments of 1972 and Section 504 of the Rehabilitation Act of 1973 or other federal, state, or local laws or executive orders.

In addition, Carnegie Mellon University does not discriminate in admission, employment or administration of its programs on the basis of religion, creed, ancestry, belief, age, veteran status, sexual orientation or in violation of federal, state, or local laws or executive orders. However, in the judgment of the Carnegie Mellon Human Relations Commission, the Department of Defense policy of, "Don't ask, don't tell, don't pursue," excludes openly gay, lesbian and bisexual students from receiving ROTC scholarships or serving in the military. Nevertheless, all ROTC classes at Carnegie Mellon University are available to all students.

Inquiries concerning application of these statements should be directed to the Provost, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, telephone (412) 268-6684 or the Vice President for Enrollment, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, telephone (412) 268-2056.

Obtain general information about Carnegie Mellon University by calling (412) 268-2000.

Special Report
CMU/SEI-97-SR-019
November 1997

Workshop on COTS-Based Systems



Patricia A. Oberndorf

Lisa Brownsword

Ed Morris

Carol Sledge

(Editors)

COTS-Based Systems

DTIC QUALITY INSPECTED 4

Unlimited distribution subject to the copyright

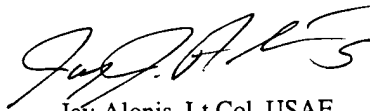
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

This report was prepared for the

SEI Joint Program Office
HQ ESC/AXS
5 Eglin Street
Hanscom AFB, MA 01731-2116

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

FOR THE COMMANDER



Jay Alonis, Lt Col, USAF
SEI Joint Program Office

This work is sponsored by the U.S. Department of Defense.

Copyright © 1997 by Carnegie Mellon University.

Permission to reproduce this document and to prepare derivative works from this document for internal use is granted, provided the copyright and "No Warranty" statements are included with all reproductions and derivative works.

Requests for permission to reproduce this document or to prepare derivative works of this document for external and commercial use should be addressed to the SEI Licensing Agent.

NO WARRANTY

THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

This work was created in the performance of Federal Government Contract Number F19628-95-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 52.227-7013.

This document is available through Asset Source for Software Engineering Technology (ASSET): 1350 Earl L. Core Road; PO Box 3305; Morgantown, West Virginia 26505 / Phone:—(304) 284-9000 / FAX—(304)-284-9001 World Wide Web: <http://www.asset.com> / e-mail: sei@asset.com

Copies of this document are available through the National Technical Information Service (NTIS). For information on ordering, please contact NTIS directly: National Technical Information Service, U.S. Department of Commerce, Springfield, VA 22161. Phone—(703) 487-4600.

This document is also available through the Defense Technical Information Center (DTIC). DTIC provides access to and transfer of scientific and technical information for DoD personnel, DoD contractors and potential contractors, and other U.S. Government agency personnel and their contractors. To obtain a copy, please contact DTIC directly: Defense Technical Information Center / Attn: BRR / 8725 John J. Kingman Road / Suite 0944 / Ft. Belvoir, VA 22060-6218 / Phone—(703) 767-8274 or toll-free in the U.S.—1-800 225-3842.

Use of any trademarks in this report is not intended in any way to infringe on the rights of the trademark holder.

Table of Contents

1. Introduction	1
1.1 The First COTS-Based Systems Workshop	1
1.2 Participants	1
1.3 Workshop Structure	1
1.4 Structure of This Report.....	2
2. Workshop Results and Themes.....	3
2.1 Future Vision.....	3
2.2 Recurring Workshop Themes	4
2.2.1 Evaluation Is Continuous, Not a Discrete Step	4
2.2.2 Intimate Relationship of Evaluation and Design.....	4
2.2.3 Influence of the Marketplace on Requirements	5
2.2.4 CBS Approach as Risk Mitigation.....	5
2.2.5 Importance of Prototyping	5
2.2.6 New Skill Sets Needed	5
3. Technology and Product Evaluation Breakout Group	7
3.1 Why Do We Evaluate?	7
3.2 What Do We Evaluate?.....	9
3.3 How Do We Evaluate?	10
3.4 Why Is Evaluation Hard? What Can We Do to Make It Easier?	13
3.4.1 Evaluation Case Studies	14
3.4.2 Lessons Learned from the Case Studies.....	18
3.4.3 Things We Need.....	19
3.4.4 Do's and Don'ts	19
3.4.5 Risk Management	20

3.5 Conclusions of the Technology and Product Evaluation BOG	21
4. Engineering and Design Breakout Group	23
4.1 Inception Phase Issues.....	24
4.2 Elaboration Phase Issues	25
4.3 Construction Phase Issues	29
4.3.1 Construction Techniques.....	30
4.4 Transition Phase Issues	31
4.5 Evolution Phase Issues	32
4.6 Conclusions of the Engineering and Design BOG.....	33
4.6.1 Problem Summaries.....	33
4.6.2 Observations.....	35
5. Acquisition and Management Breakout Group	37
5.1 Metrics.....	39
5.1.1 Why Should I Measure?	39
5.1.2 What Should I Measure?	40
5.1.3 How Do I Measure?	40
5.1.4 What Are the Tradeoffs?	42
5.2 Wisdom and Understanding.....	42
5.3 Policy and Guidance.....	44
5.3.1 Guidance for Contracting	44
5.3.2 Guidance Regarding Standards	45
5.3.3 Guidance Regarding Sustainment.....	45
5.4 “Answers”	47
5.5 Conclusions of the Acquisition and Management BOG.....	48
6. Workshop Conclusion	49
Appendix A: Workshop Participants	51

A.1 Technology and Product Evaluation Breakout Group	51
A.2 Engineering and Design Breakout Group	51
A.3 Acquisition and Management Breakout Group	52
Appendix B: Unified Modeling Language	53
Appendix C: The Five-Panel Model	55
Appendix D: List of Acronyms	57
References	59

List of Tables

Table 1. Some Things to Measure

41

List of Figures

Figure 1: Progressive Filtering	10
Figure 2: Process Tool Selection	14
Figure 3. Strategy Analysis Matrix	47
Figure 4: The Five-Panel Model	55

Report on the First COTS-Based Systems Workshop

Abstract: This report documents the proceedings of the first Workshop on COTS-Based Systems, held at the Software Engineering Institute (SEI) June 10-11, 1997. It describes the workshop activities, the discussions of three breakout groups, and some general conclusions reached by participants in the workshop.

1. Introduction

1.1 The First COTS-Based Systems Workshop

The first COTS (commercial off-the-shelf)-Based Systems (CBS) Workshop was held in Pittsburgh, Pa., at the SEI on June 10 and 11, 1997. The workshop brought together practitioners of various aspects of COTS-based systems management, procurement, development, and maintenance in order to share the problems that must be faced and the solutions that they have found.

1.2 Participants

The participants in this workshop represented many different organizations and brought with them a wide variety of experiences. Participants included representatives of the U.S. Department of Defense (both civilian and military), other civilian federal agencies, large software systems organizations, other federally funded research and development centers (FFRDCs), and academia. Participants from the National Research Council of Canada provided an international perspective. The workshop was hosted by the Dynamic Systems Program of the SEI, but several other SEI programs participated as well, contributing their particular expertise. Participants also came from the Carnegie Mellon University campus.

Appendix A provides a complete list of participants, arranged according to the breakout group in which they participated.

1.3 Workshop Structure

The workshop began with a plenary session, during which Tricia Oberndorf (SEI) discussed the goals and structure of the workshop and some foundational ideas from current SEI work in the area of COTS-based systems. This served to create a common perspective for participants so that subsequent sessions could be more productive.

After the plenary session, the workshop participants joined one of three breakout groups (BOGs):

- Technology and Product Evaluation
- Engineering and Design
- Acquisition and Management

The charter of each group was to identify CBS issues in their topic area, elaborate on the critical issues, and identify possible solutions (e.g., approaches, sources). The task before them was to capture the participants' experiences and the issues they have been facing in dealing with CBS, then to discover what approaches have been used and to identify sources for further information.

Over the course of the two days, the groups had wide-ranging discussions of a number of issues regarding COTS- and component-based systems. The breakout groups reported their findings back in plenary at the end of the second day of the workshop.

1.4 Structure of This Report

The remainder of this report documents the results of this workshop. Chapter 2 presents the results of a brainstorming session in which participants contributed to a future vision of COTS-based systems. The sections in Chapter 2 capture recurring themes that emerged from the workshop discussions. Chapters 3 through 5 present the results of the three breakout groups (BOGs):

- Technology and Product Evaluation (Chapter 3)
- Engineering and Design (Chapter 4)
- Acquisition and Management (Chapter 5)

Chapter 6 presents a short conclusion. The appendices list the participants in the three BOGs (Appendix A), describe the unified modeling language (Appendix B), provide an overview of the five-panel model (Appendix C), and define the acronyms used in this report (Appendix D).

Throughout this report, the editors attempted to present the information exactly as it was recorded during the workshop, in order to ensure that we did not change the meaning of the findings. As a result, the wording of some of the findings and conclusions in this report may not always flow smoothly.

2. Workshop Results and Themes

2.1 Future Vision

During the initial plenary session, workshop participants were asked to help build a vision of software and system engineering based on disciplined practices for the use of COTS components. Elements of this vision that emerged during the discussion include

- different life cycles
- processes and tools for evaluating components
- new maintenance approaches
- a new or revised Capability Maturity Model SM (CMM[®])
- better knowledge of CBS risks and tradeoffs, plus a method for measuring them
- improved ways of defining requirements that simplify component evaluation
- appropriate standards for hardware, software, processes, and interfaces
- a better understanding of the central role of architecture
- recognition of the fluidity of COTS-based system design
- better understanding of the effect of COTS on system architecture, design, safety and security
- more complete and accurate component specifications
- a catalog or repository of component information, with sufficient resources to maintain it
- fixed architectures and component stores for some domains
- a new industry where integrators bundle together components into larger packages
- a framework for testing multiple components working together
- new laws fostering competition
- modified Federal Acquisition Regulations (FARs) to better encourage the use of COTS
- new laws concerning liability
- updated license management and payment schemes
- a well-defined business case
- proactive incentive and management
- risk mitigation strategies
- strategic alliances with vendors

SM Capability Maturity Model is a service mark of Carnegie Mellon University.

[®] CMM is registered in the US Patent and Trademark Office.

- definitions for the new skill sets required
- better educated engineers, managers, and customers

As the workshop proceeded, these ideas re-emerged in various forms from the three breakout groups.

2.2 Recurring Workshop Themes

During the course of the workshop, different sessions and BOGs focused on a wide variety of topics. However, certain themes emerged repeatedly, in different groups at different times, and were arrived at for different reasons. The most important of these themes are discussed in Sections 2.2.1-2.2.6.

2.2.1 Evaluation Is Continuous, Not a Discrete Step

One theme that emerged clearly from all three breakout groups was the realization that the evaluation of software components must become a continuous process during the lifetime of a COTS-based system. That is, evaluation is not just something that is done to get such a system started; rather, it is something that is done continuously, since the updated information will be continuously needed until system retirement. This continuous evaluation not only reflects the upgrade cycles of component vendors; it also provides a way of identifying new technologies that can support the evolution of a system.

2.2.2 Intimate Relationship of Evaluation and Design

In connection with the above observation, the workshop participants noted that both system requirements and the marketplace are changing. The activities of evaluation serve to bridge the gaps between them and so become closely intermingled with the design of the system. It is tempting to make the mistake of thinking that evaluation is a separate activity from design. But if these activities are not intertwined, system designers will find themselves specifying systems that cannot be built with commercially available components or failing to make use of particularly promising capabilities emerging in the marketplace.

In this scenario, evaluation becomes a part of design and engineering and also of management and acquisition, making it impossible to truly separate these activities into distinct life-cycle phases. This impact of the marketplace on design has profound implications for new CBS life-cycle models and processes. Unfortunately, the participants in the workshop were not able to explore the evolving CBS life-cycle models and processes due to lack of time.

2.2.3 Influence of the Marketplace on Requirements

Another closely related theme recognizes another impact of the marketplace on the development and evolution of COTS-based systems: the requirements need to be much more malleable than with traditional system approaches. A strictly top-down systems approach (freeze the requirements, then freeze the design, then implement the system) did not work well in the past because it was not possible to anticipate how the project would progress or change over a long development period. Now, this process will not work at all for COTS-based systems because the very thing on which these systems will be based—the COTS marketplace—is in constant change. The key to sound COTS-based systems development in the future will be well thought-out requirements that are clearly distinguished and prioritized. The prioritization will need to recognize which requirements are absolutely essential, which are needed but might be achieved in a variety of ways, and which would just be “nice to have.”

2.2.4 CBS Approach as Risk Mitigation

All three breakout groups identified approaches to CBS in terms of risk mitigations. In other words, the things that need to be done differently for CBS are largely targeted towards addressing the potential risks introduced by the use of commercial products in mission-critical systems.

2.2.5 Importance of Prototyping

One can never be certain of all of the hidden assumptions embedded in a given product. Since CBS are dependent on commercial and other non-developmental products, the characteristics of a system built from such products will also be uncertain. This increases the need for prototyping the system to identify the responses of the individual products and of the entire system under a range of operating conditions.

2.2.6 New Skill Sets Needed

All of these things taken together emphasize that the development of CBS is a new way of doing business in the development of systems. It is common for higher level managers and policy makers to surmise that using commercial products and depending on commercial industry will reduce the need for government expertise. All three breakout groups came to

exactly the opposite conclusion. The dependence on commercial industry and products *increases* the need for the government to be a smarter consumer. This in turn takes *more* expertise, not less. The bottom line is that using COTS is harder, not easier.

3. Technology and Product Evaluation Breakout Group

Members of the Technology and Product Evaluation BOG represented industry, government, and academic perspectives. Appendix A lists the members of the BOG.

This BOG overcame the interplay between multiple personalities (two facilitators!) to identify a number of metaphors for CBS evaluation strategies. They worked through several instances of evaluations described by participants, from which some themes and ideas emerged. In addition, the breakout group identified a number of evaluation “dos” and “don’ts.”

This breakout group structured its discussions around a series of important evaluation questions:

1. Why do we evaluate things? (See Section 3.1.)
2. What kinds of things do we evaluate? (See Section 3.2.)
3. How do we evaluate things? (See Section 3.3.)
4. What is hard about evaluation, and what do we do to make it easier? (See Section 3.4.)

The sections that follow cover these questions.

3.1 Why Do We Evaluate?

After some initial discussion, it became clear that different individuals had different ideas of why we need to evaluate software. The discussion concentrated on three forms of evaluation:

- evaluation of a product against what it is claimed to do
- evaluation for the qualities we would like in a particular system
- evaluation of fitness for use in a given context

We often want to measure (evaluate) whether a component actually provides the services that it claims to provide. Sometimes these services are only documented in a user manual or a brochure. In other cases, these services are documented in a published interface document (such as a published application program interface [API]) or component specification. In still other cases, the specification against which the component is evaluated is a *de facto* or formal standard; this form of evaluation is sometimes called validation or conformance testing. A closely related activity often involves determining whether the services provided are actually the ones we want. That is, not only does the component do what it says, but are these the things that we want it to do?

A second reason we evaluate is to determine whether the component demonstrates the qualities that are necessary for the system, but are not necessarily represented in the

interface. The BOG participants suggested that such “non-functional” qualities as reliability, maintainability, security, and performance are particularly hard to determine in a COTS component, yet have a huge impact on the viability of the component and the overall system. Evaluation of non-functional qualities is particularly hard because we often have an incomplete understanding of non-functional requirements, coupled with inadequate tools and techniques for determining whether the software meets expectations.

After some discussion, the BOG recognized that “fitness for use” within a given context encompassed the other two forms of evaluation, along with many other technical, organizational, business, and political factors. There are two important points here. First, the fitness for use of a component is the product of many factors, and an attempt to evaluate a component from a single viewpoint will likely fail. For example, even a component that is extremely well suited from a technical standpoint will not be fit for use if there are strong political trends that work against it. Second, we can evaluate the fitness of a component only within a specific (and concrete) context. To perform a meaningful evaluation, we must have a purpose, a set of requirements, and criteria for determining whether a component meets those requirements. Without this context, evaluation is not meaningful.

The role of context in determining fitness for use was made clear to the BOG when one of the facilitators suggested they discuss the evaluation of two distributed object technologies (Common Object Request Broker Architecture [CORBA] and Distributed Component Object Model [DCOM]). After a few minutes in a different sort of bog, the group came to the conclusion that without a meaningful (i.e., real) context in which to perform such an evaluation, the exercise was meaningless. This conclusion has far-reaching ramifications for organizations intent on developing “pre-approved” lists of components. At best, such lists can provide only very general guidance by eliminating grossly flawed components from consideration. At worst, a list of pre-approved components can misdirect an organization into a false sense of security regarding approved components, since inclusion on such a list does not ensure fitness for use in a specific system situation.

This early discussion also hinted at a different role for component evaluation. Various BOG members suggested that, in the course of an evaluation, we often make serendipitous discoveries that alter the perceived architecture or design of the system, or even the system requirements. This suggests that evaluation and selection of components may play a role in virtually all phases of system construction. In other words, evaluation cannot be divorced from system specification, architecture, design, and implementation. This will have strong ramifications for the processes that must be developed for creating COTS-based systems.

The members of the BOG recognized that, regardless of the motivation, evaluation is a decision *tool*; it is *not* the decision. Evaluation is not an end in itself but is designed to support certain kinds of decisions. Such decisions include

- determining fitness for use of those aspects of the component that we desire
- make-buy decisions
- choosing among products

- choosing among vendors
- discovering properties of the system
- validating claims
- making architectural, design, and requirement decisions

Often such “smaller” decisions support the major decisions necessary for a project to meet cost and schedule goals.

3.2 What Do We Evaluate?

Products, vendors, customer bases, and activities involved in rolling out products (support, training, etc.) are among the more obvious things identified as potential subjects of evaluation. A partial list of other subjects of evaluation efforts includes

- a technology, as opposed to a product (for example, CORBA technology vs. a CORBA implementation from a specific vendor). However, technology evaluation can be costly, and there are often many unknowns about what problem the technology will solve and how the technology will evolve.
- the maturity of a technology and of vendors
- the long-term viability of the product or technology
- product design patterns, which can affect the resources of the system and its evolution
- the architectural model of a product
- the impact of a product or technology on the computing paradigm
- a specific subset of a product or technology
- the qualifications of the vendor
- the ability of the organization to adopt and manage the product

The BOG repeatedly reached the conclusion that evaluation cannot be focused on only a few characteristics of a product. Rather, the evaluation process will be multifaceted, with many different product, vendor, technology, organizational, and other factors considered.

The BOG also recognized that evaluation is not a “one-time” occurrence. Evaluation of components occurs during (at least) three distinct activities during the system life cycle:

- when making choices of technologies, products, and vendors during system selection, i.e., *component selection*
- when evaluating new releases of existing components, i.e., *release acceptance*
- when evolving a system with technology or product refreshment, i.e., *system maintenance or change*

As a consequence, an organization should build up and maintain a capability to evaluate products and technologies.

3.3 How Do We Evaluate?

The next issues considered by the BOG were the ways in which we evaluate COTS components and whether we evaluate different kinds of components differently.

While a number of different approaches for COTS product and technology evaluation were discussed, we can summarize the work of the BOG by postulating that the evaluation approaches discussed employ three basic strategies:

- progressive filtering
- puzzle assembly
- keystone identification

Progressive filtering

Progressive filtering represents a strategy whereby a component is selected from a larger set of potential components. Starting with a candidate set of components, progressively more discriminating evaluation mechanisms are applied in order to eliminate less fit components. Early filters facilitate rapid judgments based on information provided directly by the component vendor (e.g., specifications, component documentation). Intermediate filters may involve actual execution of the component (often in a stand-alone mode) to further assess suitability. Later filters often involve integration and execution of selected component features in a testbed representing the rest of the system. Figure 1 shows the progressive filtering strategy with example filters.

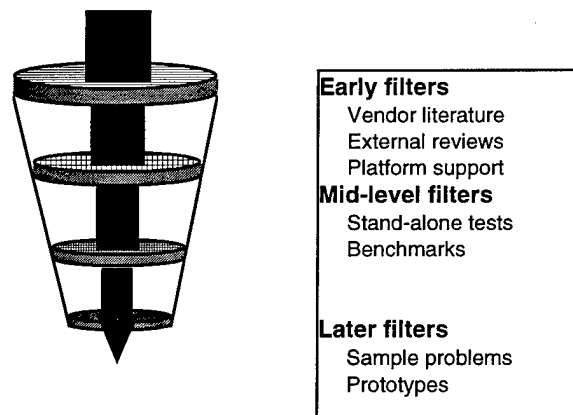


Figure 1: Progressive Filtering

In general, as the discriminatory power of the filter increases (i.e., becomes more similar to the execution demands of the developed system), the number of components subjected to the filter becomes smaller. More discriminating filters normally require far more effort to develop and use. However, they also greatly increase confidence that an appropriate

component is selected (and that inappropriate components are eliminated). The strength of the progressive filtering approach is that it tends to economize effort through low-cost elimination of some contenders, coupled with a more thorough evaluation of those that are better suited to the system.

One example of the progressive filtering technique can be found in [Lichota 97], where the set of potential components is winnowed by applying increasingly exacting evaluation techniques. After the products are identified, the initial screening is provided through analysis of product documentation, specifications, briefings, and demonstrations. Those products that survive the initial screening are then subjected to increasingly exacting evaluations involving stand-alone testing, integration testing, and finally field testing. Each round of evaluations can potentially reduce the set of components to be tested for the next round. Reducing the number of components under consideration is important, since successive evaluation phases require increasing amounts of effort. In addition, the approach provides increased confidence that the components that survive the process are likely to fulfill expectations and successfully serve in the completed system.

A discussion of evaluation case studies by members of the BOG highlighted an important consideration for the progressive filtering approach: it has little to say about the criteria that should be considered when evaluating components, or about the number, complexity, and granularity of the filters that must be applied. Different organizations employed different filters, depending on the specific context of the evaluation activity. There is no predetermined set of filters that will universally determine whether a product is fit for use.

BOG members also recognized that, in its basic form, the progressive filtering approach does not address the necessary balance between the mission, marketplace, and system during the evaluation process. However, many progressive filtering strategies discussed in the BOG's case studies included one or more feedback loops to address this problem.

Puzzle assembly

A second strategy identified by the BOG is called *puzzle assembly*. The puzzle assembly model begins with the premise that a valid COTS solution will require fitting the various components of the system together as a puzzle. That is, it is difficult to determine the characteristics of any one piece without simultaneously considering the range of characteristics of the other pieces. This implies that, just because a component "fits" in isolation (perhaps as determined by filtering techniques), it is not necessarily true that it fits when combined into a system. In addition, creating a COTS-based solution requires maintaining the malleability of requirements for the system, the components, and the interfaces until the outline of the completed system emerges. Thus, puzzle assembly involves discovery of system potential.

The puzzle assembly approach applies an evolutionary prototyping technique to build versions that are progressively closer to the final system. The extensive use of prototyping provides an opportunity to understand the characteristics of various components as well as the system. During prototyping, it is possible to get a sense of the effort required to lash together components into a system. Data concerning this effort is an important input in determining whether a component is a viable piece of the final solution.

If the pure progressive filtering technique can be thought of as a “reductionist” approach to determining whether a COTS product will suffice (by identifying filters that eliminate inappropriate components), the puzzle assembly technique focuses on the *gestalt* of the system. In other words, underpinning the puzzle assembly approach is the philosophy that the interactions between components are preeminent in determining the characteristics of the system. Further, it is difficult (and perhaps mistaken) to assemble a system by focusing on selection of individual components.

The BOG realized that these two extremes (reductionist progressive filtering and *gestalt* puzzle assembly) are rarely applied in isolation. More discriminating filters in the progressive filtering approach usually involve some concept of how the component will work with other components. In [Lichota 97], this involves a filter requiring *in situ* demonstration of the capabilities of the component interacting with other components. Likewise, the practitioners of the puzzle assembly approach apply some progressive filtering strategies, if only to winnow the number of components involved in prototype efforts; they do not develop prototypes for every potential combination of components.

Keystone identification

A number of the BOG members suggested that, regardless of whether we complete puzzles or build filters, specific component or technology selections are critical to the design and architecture of the system and the selection of other components. In some cases, these *keystones* are identified due to specific use requirements, lack of viable alternatives, or constraints imposed by existing systems. In other cases, the importance of selecting these keystone products becomes evident during evaluation of COTS alternatives. Often, keystone products provide low-level or widely used services that are employed by other COTS components.

Selection of a keystone will normally constrain the evaluator's freedom in selecting other products, since keystone characteristics (e.g., vendor, type of technology, API) will force a focus on specific characteristics of remaining components. Often, interoperability with the keystone will become an overriding concern, effectively eliminating a large number of other products from consideration. While this may eliminate some promising products, the benefits derived from the lens provided by a keystone can reduce evaluation cost and effort.

Some members of the BOG quickly pointed out that a keystone need not be a product. It may also be a strategy for selecting components or architecting the system (e.g., open system strategy, distributed object strategy), or it may be a standard (e.g., Object Management Group [OMG] CORBA). Thus, it is possible to employ multiple levels of keystones – for example, by determining that the system will employ a distributed object strategy, centered on the CORBA standard and using IONA's Orbix products.

Selecting and/or employing a keystone does not eliminate the need for evaluation, both of the keystone itself and of the accompanying components. The need for critically examining the mission, the marketplace, and the system remains when evaluating the candidates for inclusion as a keystone and for evaluating the remaining components. As such, lists of pre-approved components that intend to provide various keystone products can offer only limited guidance and should not be considered the primary selection mechanism.

The BOG recognized that most organizations could benefit by employing all of these strategies and opportunistically switching between strategies depending on the evaluation context. For example, while pursuing a puzzle strategy, we may recognize the central importance of a particular component decision. This keystone decision point may best be addressed by progressive filtering of keystone candidates. Part of this filtering effort may involve *in situ* analysis of candidates (a puzzle assembly strategy) for the central component position in concert with other components.

3.4 Why Is Evaluation Hard? What Can We Do to Make It Easier?

The BOG quickly recognized that evaluation is hard for a variety of reasons. These include

- the many dimensions of the COTS component to be evaluated
- the lack of consensus evaluation approaches
- the subjective nature of evaluation data
- the pace of change within the COTS marketplace
- the lack of visibility into the internal workings of the product

Many other reasons were suggested as well. In fact, what seemed to motivate participation in this particular BOG were previous difficulties in making good evaluation decisions. The BOG mined these experiences by encouraging participants to discuss their problems with and solutions for evaluating COTS components. Some of these case studies appear in the following sections.

3.4.1 Evaluation Case Studies

3.4.1.1 Selecting a Process Tool

One participant spoke of his experiences selecting a process tool. This evaluation activity took about three weeks of effort and is illustrated in Figure 2. First, he gathered requirements from the customer, based largely on his knowledge of software processes and hypothetical users. In parallel, he performed a rapid, partial market survey largely from the Web, tool fairs, cross references he obtained, and the like. This market survey yielded feature lists and candidate literature, which were brought together with the requirements for an initial screening.

The initial screening reduced the number of candidates; the remaining ones were then subjected to some scenario-based testing, using evaluation copies of the products. A final test on the full product resulted in a suggestion to the customer for the tool to choose.

The participant noted that he did not attempt to qualify the vendor (which was justified by the project timeline and scale of the effort) and he collected no testimonials from users.

The BOG member who performed this evaluation activity noted that he was very disappointed with the outcome, since his work achieved little visibility within the target organization.

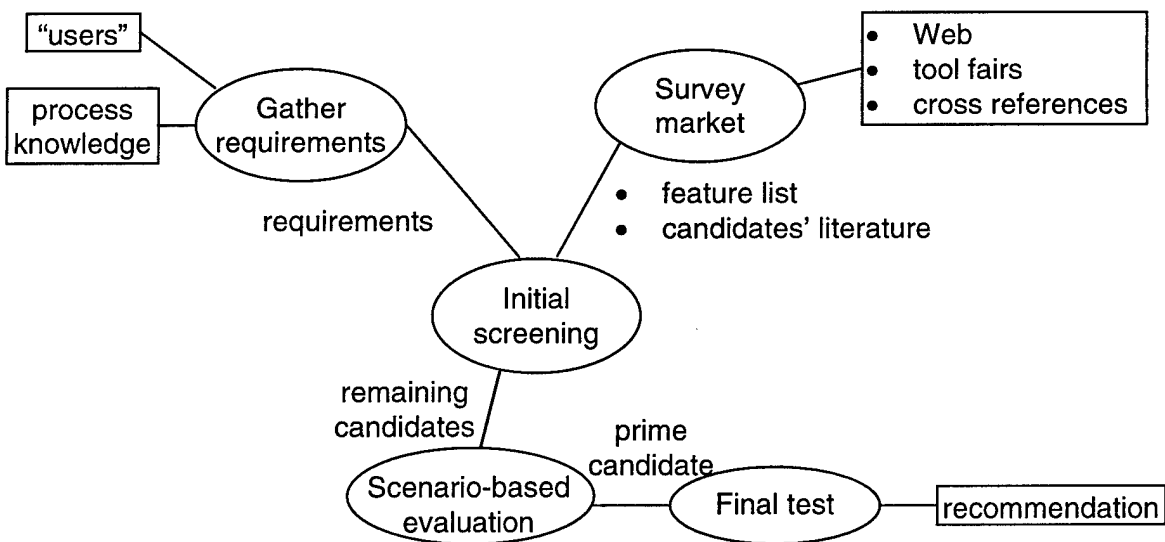


Figure 2: Process Tool Selection

3.4.1.2 Image Delivery Across a Network

Another participant reported on an effort to enhance an existing database encased within an inadequate system. The upgraded capability should handle 20 million large images, scaled between 100K and 4M bytes. This database was to be accessed from widely dispersed (worldwide) sites. The evaluation involved identifying a suite of products that could be integrated to support this overall scenario.

Requirements were gathered as use-case scenarios. Groups of products were identified and classified (e.g., databases, user-interfaces, web interfaces, image manipulation products). The effort avoided describing any particular architecture; the components were expected to drive the architecture.

To find candidates, the evaluators searched the Web and attended product fairs. A small number of candidates in each class was chosen for further review, based on the initial screening of products. Then the customer decided they wanted an NT platform, at least for the "server." This almost forced the choice of ODBC (Open Data Base Connectivity), rather than dealing with databases directly.

Following the initial screening, the evaluators built some basic prototypes, including a Web browser and server, using Internet Explorer. During prototyping they discovered that the available version of Internet Explorer could not execute the "POST" command (a feature that Explorer is advertised to have), so Netscape Navigator became the browser of choice.

At this point the evaluators began to consider various architectural alternatives. No anchor product (keystone) had been identified at the time of this workshop. However, prototyping had yielded a system capable of using four different Web servers without changes to the code. Thus they are plug compatible in this respect.

In order to screen candidates, the evaluators read testimonials, but they did not do vendor qualification. A painful lesson learned was that you cannot always believe the marketing hype. They have also not done comprehensive testing of the components, nor have they looked at the business case—that will be the sponsor's job. The existing prototype is intended to generate the requirements for the request for proposal (RFP) for the actual system.

3.4.1.3 Procurement of Integrated CASE Environment

In this case study, the evaluators began with three sets of draft requirements from the customer. Based on these requirements, the evaluators developed an initial design concept and an initial categorization for the required tools. They gathered information on the various tools within these categories, largely from product literature and other free information. Initially they picked two to three tools in each category.

Part of the evaluation process was to construct "compliance matrices" comparing tool features to requirements. The screening process included a check for certain important

features (e.g., integrability, availability of an API, verification that access was possible, and some very simple prototypes), as well as vendor presentation demonstrations and full evaluation copies. Vendors were invited to give in-house demonstrations and then the evaluators received evaluation copies. Evaluators tried to insist (mainly successfully) on complete functionality from vendors, rather than crippled versions of tools. Most tool evaluation copies were free. An important criterion for vendor qualification was vendor “cooperativeness,” and the evaluation process provided a good opportunity to assess the developing relationships with vendors. The evaluators also looked for the ability to access data within the tool. Somewhat surprisingly, the vendor demonstrations were particularly effective at screening out candidates, since a number of vendors did not prepare adequately to demonstrate their tool in its best light.

Some anchor (keystone) products were identified first. The evaluators then considered how well other tools integrated with the anchors. The anchors chosen were the “normal” computer-aided software engineering (CASE) tools. This was often a “gut-feel” decision.

Subsequent to tool selection, the team began to design the repository and integrate tools. They found that prototype tests, scripts, and wrappers that had been developed were often not adequate to address the problems that arose. Environmental conflicts between tools were a big source of problems, but most were not discovered until the wrapping and integrating activity began. Another major issue involved the differing semantics used by various tools – this made data integration difficult.

The evaluators did gamble a bit with some products, based on enhancement plans promised by vendors rather than actual product features. As a result, the evaluators were “burned” on two of five selections, but they did have fallback positions for these instances. They ended up purchasing 60 different products from 23 different vendors (although it is hard to count the number of products, since it depends on how they are packaged or billed).

3.4.1.4 Getting Corporate Buy-In for a Requirements Management Tool Across a Corporation

The challenge for this effort was to choose a requirements management tool to be used across the corporation—about 50,000 people in 25 locations. The first issue was formulating the selection team, which consisted of a systems engineer and a software engineer from each of the four operating sections of the corporation, thus guaranteeing selection by committee. In all there were about a dozen people involved in the selection process.

The selection committee narrowed the original field of 23 candidates down to 7, based on brochures and what people knew about specific tools. This screening process was *ad hoc* and was not based on system requirements. A request for information (RFI) was issued to the surviving seven vendors, and a set of questions was sent to each. Based on answers received from all seven, the set was further narrowed to four.

The four surviving tool vendors were asked to provide a half-day demonstration of their tool. The first half of this period was to be a general, undirected demonstration of the tool's capabilities; the second half involved a directed demonstration, with the evaluators asking to see specific tool actions. Some vendors had nothing prepared to show in the undirected half. On the basis of these demonstrations, the field was further narrowed to three. This was further narrowed to two based on vendor qualification, taking into account such things as market share, revenues, and stability. One candidate was a "mom-and-pop shop," and the evaluation committee had no confidence that they could meet a major commitment to a large corporation. The evaluators also asked questions about market share, and the answers for this set of vendors added up to somewhat more than 100%.

A request for quote (RFQ) was sent to the remaining two vendors, now constrained by legal considerations as well. This document addressed about 40 major considerations. Some considerations (e.g., ease of use) were supported by an evaluation matrix with ranges of points. The subjective nature of this approach was highlighted when some evaluators tried to jockey the evaluation criteria to support their viewpoints. In other cases, evaluators proved to be quite naïve concerning the real capabilities of tools.

The group (officially) did not use any of the tools or perform any hands-on evaluation; they relied on proxy experience. They did interview other users, and requirements for the selection were adjusted through the process.

The end result was the choice of a tool. However, instead of the expected large procurement, only 150 licenses were obtained. Approximately one dozen projects are using the tool today. One roadblock to wider use was that management did not encourage or fund the deployment of the selected tool.

3.4.1.5 NASA COTS Tools Versus Components

This case study involves a make-or-buy decision for building flight dynamics software. Each satellite launched needs this flight software, and five or six satellites are launched per year. Strong advocacy groups supported both make and buy decisions. One group advocated the use of a commercially available set of libraries, while a second group wanted to continue to use an existing C++ component library. Both groups focused on a specific advantage of their preferred solution; the group advocating the COTS solution desired the reliability of a COTS product, while the C++ group favored maintaining control over changes and fixes.

The compromise evaluation strategy involved the groups each using their alternative to build the necessary software, and then comparing and contrasting the results. The primary evaluation criteria were cost, schedule, and reliability. Unfortunately, the biases of the groups interfered with the evaluation strategy. The groups employed their preferred approach, and only one group (the one favoring C++) collected the metrics necessary to make a sound decision. In the end, each group concluded they were right and declared victory.

3.4.1.6 Choosing an Email System

In this case study, an organization made a decision to hire a consultant to choose an email system. Unfortunately, the chosen consultant had a vested interest in a specific solution. The consultant recommended his preferred solution, but was ignored by the organization. Thus, the money spent on the consultant was wasted. The lesson here suggests that efforts should be made to remove sources of bias from any evaluation.

3.4.2 Lessons Learned from the Case Studies

The BOG made some observations that constitute lessons learned from these case studies:

- There is no single, "one size fits all" evaluation process.
- An early data-gathering and discovery stage was important in all case studies. The complexity and formality of this stage varied widely.
- All case studies were based on a process of elimination and employed some version of progressive filtering. Some also employed elements of puzzle assembly and keystone identification.

The BOG also drew some additional conclusions about what is hard about evaluation and why:

- There are many existing and new components and versions with many interactions. Management of components themselves becomes difficult as new versions are delivered.
- There is often excess marketing hype and a corresponding lack of engineering information. Some vendors appear to be incapable of anything other than hype.
- There are many undocumented system and environmental assumptions on the part of the product makers. These often lead to conflicts when multiple components are integrated.
- There is a lack of qualified evaluators—a new skill set is needed.
- There is often a lack of sufficient time and resources for performing evaluations. Not enough emphasis is placed on the evaluation activity. The identification of an accepted process model can help remedy this situation.
- There are many dimensions of the vendor and product to measure but a lack of measurement techniques.
- One of the hardest problems is predicting where a technology or a product will be years down the road, as the system is being maintained. As Neils Bohr said, "Prediction is difficult, especially the future."

3.4.3 Things We Need

The BOG generated several lists of processes, tools, and techniques that must be improved to support a CBS approach. The following collapses and summarizes these lists:

- appropriate reference models for COTS products. It is difficult to compare products, since there may be no common vocabulary or taxonomy. A reference model could be an important instrument for evaluation.
- better support for prototyping and simulations, including benchmarking, testing, probing, and performance tools
- better sources for market research data. It can be difficult to obtain accurate data—consider that vendor-supplied market shares have been known to cumulatively exceed 100% of the market!
- improved methods of evaluating vendors. Current criteria, such as time in business, responsiveness to customers, and willingness to back their product, are useful where a documented track record is available. However, as a component market develops, and components become “smaller” (like Java Beans or Active-X components), they are increasingly coming from small vendors. This can make evaluation of vendors more difficult.
- better understanding of business factors, such as licensing, business case analysis, and loaded costs of COTS use over the lifetime of the system
- COTS risk classification and analysis approaches that address unknowns, confidence levels in analysis, and the impact of inappropriate decisions
- better methods for documenting the context (i.e., system architectures and interfaces) in which the evaluation is occurring and the expectations placed on the component by the rest of the system

3.4.4 Do's and Don'ts

The BOG then agreed to try to formalize some of the things that should—and should not—be done in evaluations.

Do

- Guard against prejudice in the evaluation activity. Prejudice crept into a number of the activities, leading to poor decisions or no decision in some cases.
- Create flexible requirements. In a number of cases, evaluators found better and/or cheaper ways to do things when afforded flexibility in system requirements. They were able to use COTS components to their best advantage in these cases.
- Evaluate your components in context. Stand-alone evaluation of components proved inadequate except when the component was to be used alone. Prototypes that combine multiple system components and evaluation via operational scenarios appear to be essential techniques.
- Plan for cost and schedule variances. The existing cost models are probably inappropriate for COTS-based system development.

- Assess necessary and sufficient parts, but do not “gold plate” requirements. Gold plating of requirements, while tempting, will likely lead to a “build” rather than a “buy” decision. Common experience of the BOG suggests that gold plating rarely leads to a better system anyway.
- Use COTS where appropriate. This implies that COTS is inappropriate in some cases. Thus, do not be blindly led by policies or pronouncements that mandate a certain level of COTS. The primary weapon of the evaluator when arguing against a stated policy is a well-conceived and thorough COTS evaluation.

Don't

- Make someone else responsible for your evaluation. While a contractor can bring important capabilities to the table, you must live with the decision. In addition, the knowledge you gain about components during evaluation is critical in system design.
- Believe what you read, regardless of the source: you have to evaluate it. Thus, do not rely on the vendor's word or on lists of pre-qualified components.
- Expect 100% certainty from evaluations or 100% confidence in the results. The goal of your evaluation process should be to select evaluation methods that reduce your uncertainty (and risk) to an acceptable level.
- Go for the latest without a fallback. New technologies and products have their own set of risks that are often the result of the marketplace – and you are not big enough to drive the marketplace.
- Begin a CBS development effort with an existing (traditional software development) mindset–stay open to changes in the process model.
- Forget how much glue you need. Integrating COTS components often requires more glue than we like, and you will remember this statement every time a new release arrives.
- Assume management understands COTS issues. If we don't, how can they?
- Forget to test. An evaluation is not a system test, nor are the promises of the vendor. COTS components will likely require new strategies and extensive testing effort.
- Evaluate just for the sake of evaluation. Evaluation requires a context.

3.4.5 Risk Management

The BOG discussed evaluation as part of a risk management approach. In effect, the criteria developed for evaluating components should reflect risks concerning the components. A corollary is that if there is no risk, then there should be no criteria. While this seems like a straightforward statement, it does hint at a mechanism whereby we can evaluate criteria. The BOG went on to identify a number of particularly troublesome risks:

- Discovering faults in black boxes (COTS components) is extremely hard.
- When an integrated system composed of multiple components fails, it is sometimes hard to identify the culprit. The culprit could be any of a number of components, or the integration itself.

- Even when you identify the source of a problem, remedying or repairing faults can be extremely difficult, since you do not own the source code for the COTS components.
- Waiting for the faults to be fixed by the vendor can often result in significant delays and down time. In some cases, the vendor has little motivation to even bother with fixing the fault.
- A fix may expose (or create) another bug, forcing us to pull back for the “fixed” version of the software.
- Vendors are often motivated to envelop the older, smaller, and simpler functionality that you are using with much more expansive functionality, leading to an increasingly bloated system.
- Our own predilection is to create overly stringent requirements (or poor application of them), often resulting in the unjust rejection of potential components.

The group also discussed mechanisms to reduce the risks of incorporating a COTS component into a system. Risk reduction mechanisms include

- isolation of COTS products via wrappers and/or abstraction
- built-in logging, diagnostics, or instrumentation interfaces
- early consideration of reintegration costs that will occur during ongoing maintenance
- architecture and design decisions that increase the flexibility of the glue code in the system
- vendor qualification in terms of responsiveness and cooperation

3.5 Conclusions of the Technology and Product Evaluation BOG

The key conclusions of the Evaluation Breakout Group include

- Evaluation is intimately connected to requirements specification, design, and system architecture. This strongly suggests that the traditional waterfall model is inadequate for building COTS-based systems, since the components selected have a significant impact on the capabilities offered to the user and on the architecture and design of the system.
- No “one-size-fits-all” evaluation process is available, in large part because no two contexts for evaluation are identical.
- The evaluation situation is likely to get worse before it gets better. Our appetite for new systems does not seem to be declining. The speed of technological change and component delivery seems to get faster all the time—we have almost instantaneous delivery of some new versions of components via the Internet. This creates many problems for the maintainer of a system incorporating the component. In addition, the number of components to consider is growing rapidly as components (and vendors!) get smaller (witness Java applets and Active X components).

At the final reporting session, the Evaluation BOG presented a stunning summary of the BOG’s activities, and perhaps served to frighten workshop participants about the inadequate state of the practice of COTS evaluation and selection. However, they were encouraged by the general agreement with the conclusions – the other BOGs had developed similar positions.

4. Engineering and Design Breakout Group

Members of the Engineering and Design BOG represented industry, government, and academic perspectives. Appendix A lists the members of the BOG.

The Engineering and Design BOG defined their charter as follows:

- to identify the issues group members have encountered in developing or maintaining COTS-based systems
- to form an indication of the priorities of the issues identified
- to identify engineering and design approaches that assisted in resolving the identified issues

As the BOG discussed the scope of their charter and then throughout the ensuing identification of issues, the relationship of the topics of the three BOGs came up several times. The group agreed that the separation between evaluation, acquisition and management, and design and engineering is, in practice, artificial. There are acquisition and management aspects to most, if not all, of the design and engineering issues that each member reported. Participants also raised the need for varying kinds of evaluations throughout the life cycle.

The issues that the BOG identified as critical to the successful design and engineering of a COTS-based system centered on three themes:

1. the appropriate relationship between COTS components, requirements, and system and software architectures
2. the identification and selection of the most appropriate component integration strategy to support multiple purposes, such as instrumentation for debugging, testing, upgrading or refresh, or evolution
3. the appropriate techniques for managing the limited insight or visibility into COTS components as you select, integrate, test, and upgrade the components into the system

The BOG's discussion was structured around their charter: issue identification, candidate practices, and issue prioritization. To facilitate the identification of issues, the BOG structured its discussion by life-cycle phase. Due to the limited time for the workshop, the BOG did not attempt to define a life-cycle process for COTS-based systems. Rather, it used the phases corresponding to the development phases defined in Rational's development process [Kruchten 96]. Rational is a prominent provider of processes, services, and tools for the development and maintenance of large, complex component-based systems.

Rational's process defines the following five phases:

- **Inception:** Activities are focused on defining the scope of the system, product, or major update, including the formulation of requirements.
- **Elaboration:** Activities are focused on the selection and validation of the system and software architecture.

- **Construction:** Activities are focused on building, composing, and testing the system or product.
- **Transition:** Activities are focused on moving the system or product into operational use.
- **Evolution:** Activities are focused on continual upgrading of components due to product and technology advances, changes in mission needs, or error correction.

The following sections summarize the issues that the BOG identified for each of the above phases. While all of the issues surfaced by the BOG had engineering and management process implications, the BOG focused on the technical issues inherent in the development and evolution of COTS-based systems.

4.1 Inception Phase Issues

Activities in the inception phase focus on determining the scope of the system or major update. Typical activities would include requirements definition and analysis and the identification of key technologies and probably products. The following issues were identified by the BOG:

- *Relationship of requirements, architecture, and technology and product selection*

Like the Evaluation BOG, the Engineering and Design BOG uncovered the questions of whether the requirements drive the component selection or the available components drive the requirements, and thus the architecture. Different parts of the system may be driven by either requirements or by the availability of COTS products. When is it appropriate to let requirements drive the architecture or to allow technology, such as COTS products and standards, to drive the architecture?

One BOG member reported on a project for which requirements drove the development of previous versions of the system. Now they would characterize the project as driven by available technology. Other members of the BOG observed that, once you are technology driven, you risk being at the whim of the vendor.

- *Role of evaluation*

The BOG dealt briefly with the question of evaluation. They agreed that evaluation approaches should be “real world,” if possible. Participants gave pragmatic examples of product evaluation forms, such as multi-page checklists and weighting schemes for product characteristics of importance to a project. The group noted that the value of a checklist is that it makes you think about the important issues.

One of the challenges of evaluation is that you often find yourself looking at products that are “apples and oranges.” Reconciling these differences in order to get useful—and comparable—information from all of the products may be difficult. In addition, participants reported problems with validating a product that does not do what the vendor says it will do after delivery.

It is one thing to specify a functional interface, but the real problem in the domain of real-time and reliability-dependent systems is that the *behavior* is not well specified. The vendors do not even talk about behavior. The BOG felt it was important to discuss “cracking open” vendor products to determine their actual behavior.

- *Evaluating sets of components*

One may be faced with a choice between using a set of three products or a single other product. This emphasizes the need to prototype parts of your system with multiple components to provide an objective basis for selection.

Component evaluation needs to take place in a clearly defined context. This is complicated, however, if there are choices between sets of components. An individual component may be a driving reason behind a particular design or architectural approach. You increase the potential for mismatch with the introduction of every additional component. Part of the solution is to realize that you cannot stop after evaluating components individually; you must also evaluate together the set of components that is being considered for integration into a given system.

One member of the BOG observed that there is also a risk if you use your best people to evaluate a technology: the prototype will succeed, but the average people in your group may not be as successful in implementing a real system using the technology. Thus the usability of a given technology by your engineers should be considered in the overall technology and product selection.

- *Role of negotiation*

If customers ask for the use of COTS products, there must be a tradeoff between requirements and the degree to which COTS products can be integrated. Say you have a system that generally does X. There are often show-stopper requirements that prevent purchasing a single system to do X. A lot of what drives this is the desire to use as much available technology as possible. Both technology *and* requirements drive the architecture. What is an effective process? What considerations should such a process encompass? How should projects identify the costs, benefits, and risks, and apply this information in making requirements and technology tradeoffs? These questions point to the need for negotiation in determining the final form of the system and the extent of its dependence on COTS products.

4.2 Elaboration Phase Issues

The primary focus of the elaboration phase is the architecture. To avoid a lengthy diversion into defining an architecture, the BOG members agreed to the following broad definition:

Architecture can be defined as components and interactions. The architecture defines the components and the interactions between the components.

A number of the issues identified overlap the inception issues concerning the role of requirements, architecture, and available technology. The BOG identified the following COTS-based architecture issues:

- *Relationship of architecture to technology and product selection*

The question of how technology drives the architecture is the flip side of asking how components fit into an architecture. There emerged from the discussion the notion of a symbiotic process centered on designing an architecture based on the requirements of the system and the available components. This raised the issue of the need for ways to

work with the *set* of components under consideration, not with each individual component.

How does the available technology drive the architecture? Sometimes a component has too little functionality or too much functionality, which will only serve to further complicate these questions.

- *Relationship of architecture and components*

BOG members observed that each COTS product is built to operate with a particular architectural style [Shaw 95], such as pipe and filter or distributed system. Thus it is important to determine what is the kind and degree of match or mismatch between each COTS product, other components under consideration, and the architecture. (See [Garlan 95] for a further discussion of architectural mismatch.) This relationship may affect what architecture and what components you select. In sharing experiences, the BOG identified the following as viable mechanisms for managing the issues of architecture and component mismatch:

- standards
- component behavior specifications
- adaptation strategies
- component evaluation checklists

While adaptation approaches, such as wrappers, glue code, or bridges, can be used to compensate for the mismatch between a component and the architecture, the participants raised a related issue regarding guidelines for determining what is an appropriate amount of adaptation code. Many participants noted incidences of investing excessive resources or producing excessive code to incorporate a COTS component.

- *Specification of component behavior*

Currently there is no standard way vendors capture or describe the behavior, architectural assumptions, or intended operational use of a product. Ways to determine this information and to motivate vendors to provide such information are needed. Emerging concepts, such as Mary Shaw's *credentials* [Shaw 96], may offer partial solutions for capturing component information. Credentials consist of a triplet of data, including the attribute, the assertion, and the degree of reliability of the data. For example, the assertion that a given component cannot run in the presence of virtual memory provides the degree of reliability that this has been determined through internal testing or from system documentation. Credentials provide information in addition to the behavior and interfaces of the component.

- *Selection and evaluation of architecture*

Central to the selection and evaluation issue is determining what characteristics or criteria are important to COTS-based architectures and their relative importance to the successful delivery and use of a particular system. For example, an architecture for a COTS-based system should support the replacement of components, thus helping to address the problems of product release volatility and vendor lock. Ideally, the architecture should accommodate separable components, so that the effort required to make the change is proportional to the size of the change.

While none of the participants was aware of COTS-specific architectural evaluation approaches, several were aware of some general purpose approaches; tailoring would most likely be required. Evaluation approaches cited included the SEI scenario-based Software Architecture Assessment Method (SAAM) [Kazman 96], Lucent Technologies

architectural evaluation checklists,¹ and Rational's application of use cases to validate architectures [Kruchten 95]. [Abowd 97] provides a summary and discussion of current practice in architectural evaluation.

- *Communicating COTS-based architectures*

The BOG raised a series of related architecture issues: How should architectures, including interfaces, for COTS-based systems be identified, specified, characterized, described, and defined? Positive developments that the BOG thought were applicable, in part, included architecture description languages (ADLs). A key question is whether you can describe COTS products using some type of notation. ADLs or modeling languages allow one to describe parts of an architecture to a desired level, enabling one to reason abstractly about the system itself. Some examples of such languages cited by the group are described below:

- Wright, under development by Carnegie Mellon University, was used successfully to find defects in the specification of the modeling and simulation architecture HLA (high level architecture) under development by the Defense and Modeling Simulation Office (DMSO) [Wright 97].
- Rational's Unified Modeling Language (UML) provides a common notation and semantics for specifying, constructing, visualizing, and documenting the artifacts of software-intensive systems [Rational 97]. UML fuses the concepts of Booch, OMT (Object Modeling Technique), and OOSE (Object-Oriented System Engineering). It is targeted to a broad range of application domains including telecommunications, avionics, information technology, and concurrent, distributed systems. (See Appendix B for an overview of Rational's Unified Modeling Language.)

One of the participants shared some experience in applying OMT on the TRANSCOM Regulating and Command and Control Evacuation System (TRAC2ES) project. One problem TRAC2ES encountered in using OMT to describe the architecture was the unmanageable size of the resulting set of diagrams. Another participant familiar with UML indicated that OMT focuses on the detailed design elements of a system. Other parts of UML, particularly the Booch parts, are better suited to the architecture-level aspect of a system. This example raises the issue of getting the right technology for the problem.

- *Component integration strategy*

What are the tradeoffs that need to be considered in determining whether to create a system from a large collection of small components or from a few large components? Cadre Teamwork is a COTS product, with constituent parts, that provides CASE tool capabilities. Since it is from a single vendor, the interoperation it provides among its parts without additional effort probably makes it worth using, when compared buying the components yourself from different vendors. If you did buy the components yourself, you would need to do the system integration and testing. On the other hand, you would want to verify that the level of integration of the single vendor pieces by the supplier (in this example, Cadre) is sufficient for your application.

¹ This information is referenced in the AT&T internal report, *Best Current Practices: Software Architecture Validation* (copyright 1991). AT&T, 1993.

To understand the tradeoffs, it would be useful to have information on how components can be and have been integrated in the past.

- *Identification of iteration strategy*

Most participants thought that iterative development approaches were well suited to COTS-based systems. This raises an important issue: How do we define development increments that sufficiently factor in technical risks and customer influences? The BOG had varying opinions and experiences. One member suggested starting with the smallest, easiest, verifiable step. Another member put forth a scenario (or use-case), risk-driven approach. With this approach, work on the higher risk operational scenarios (or use-cases) or with the high-risk COTS products is done in the earliest iterations.

Another participant talked about how internal requirements change the set of COTS products that you might need. He does a lot of parallel development initially. If you cannot sit down and determine that a single system is the correct solution, you may have to use multiple development paths, investigating competing technologies. Those who try this should be aware of the potential social issues that are involved in the use of parallel paths. The different teams involved in evaluating different technologies often become competitors, since they are developing different skills, and the technology team that wins out will most likely lead the overall development effort.

- *Risk identification*

The BOG listed a number of differences in the kinds of risks for CBS as compared to traditional system development. These included

- control of the functionality and evolution of a component
- validation of the functionality, specifically, does the component do what it actually needs to do?
- frequency and degree of change of a component's functionality and APIs over time
- "make versus buy" tradeoffs and decisions

Participants suggested the need to identify key component relationships or high-risk intra-component relationships, as well as critical inter-component relationships. Using something like a technical risk analysis may help to determine areas that require prototyping and detailed evaluation. One participant ventured the idea that, in outsourcing situations such as is typical in the government sector, it is more important to have a shared risk approach between the government program office and the contractors.

- *Security*

How do you ensure the security of a system built with COTS products? Many COTS products that provide required functionality may not provide the necessary security. What strategies can you apply to maintain necessary levels of security? For example, military critical data must be encrypted and secure. Company critical information accessible on the Internet may need to be encrypted. How is control over the keys to encrypt and decrypt the information maintained when using COTS products? Does the entire system have to be at the same level of security, or is it possible to segment off some sections?

4.3 Construction Phase Issues

Activities during the construction phase activities focus on building the delivered system or update. The BOG members identified a variety of issues and shared a number of construction techniques in the areas of wrappers, debugging, and testing.

The BOG identified the following construction issues for COTS-based systems:

- *Appropriate construction of wrappers*

The BOG members noted several important concerns in designing and constructing wrappers:

- driving the wrapper design by the interface that is applicable to your application or domain rather than the type of technology that is available
- precluding the future use of emerging technologies or new products as you construct wrappers
- obsolescence of a plug-in or script used to tailor a COTS product when the vendor issues a new product release
- reliance on the internal interfaces or behavior of COTS products since these are more volatile
- use of vendor-specific features, leading to loss of portability and refreshability and to susceptibility to vendor lock.

- *Validation of vendor supplied interfaces (e.g., APIs)*

Given the limited visibility or “black box” nature of COTS components, what techniques can be used to characterize and validate the behavior of vendor-supplied interfaces? Guidelines on what kind of information you should elicit from vendor sources to use COTS products most effectively is needed. The BOG observed that interface revalidation is needed with each new release. How much revalidation is necessary is likely to differ with each COTS component, depending on the maturity of the vendor and product and the underlying technology of the component.

- *Error propagation from COTS components*

How do you establish a reliable error handling capability? How do you handle errors in a stateless environment, such as hypertext transfer protocol (HTTP)? COTS components return errors through different mechanisms, such as return codes, errors written to standard error, or exceptions. It is important for the system you are building to have a single model for identifying and handling errors. Components are created with a particular context in mind, and the system integrator has a different context. At a philosophical level you can experience errors because of the differences in philosophy. Often vendors do not anticipate how a customer will use a system, and such errors can be very difficult to address.

- *Testing and debugging COTS components and your system*

What kind of testing and debugging is necessary for COTS-based systems? The COTS components must be tested or debugged without opening the “black box.” Members of the BOG had found that current black-box debugging and testing techniques are not sufficiently robust.

4.3.1 Construction Techniques

Undoubtedly the best-known technique for the integration of COTS products is the use of wrappers. The BOG discussed three different purposes that wrappers satisfy:

- insulation

A wrapper can be something like ODBC that allows you to access multiple existing interfaces using a single API.

- adaptation

For example, in a CORBA system you may want to use a component that does not yet provide a CORBA interface, so you may define one. This provides an adapter to the existing interface in anticipation that the component will eventually provide a similar interface.

In another form of adaptation, wrappers may contain logic in addition to simply providing a different encapsulation. For example, a wrapper may contain filters and other agents that convert data into a different format.

- instrumentation

Wrappers can be used to instrument, debug, or add assertions; they can assist with load testing and scouting for side effects. One participant reported experience running a component that was running fine in a Macintosh development environment but failed when integrated and run in the real operational environment. Wrappers can assist in the detection of information vital to determining what has happened in such a situation.

One technique for building wrappers is to apply abstraction concepts in designing the interface. Perhaps you as an integrator want to insulate the remainder of your system from changes to a particular component. If standard interfaces exist, it may be acceptable for you to use them to achieve this insulation; if not, then it may make sense to develop this encapsulation. If, for example, the client screens now have no knowledge of the underlying database, then this makes the system more amenable to changes in the database product.

Another technique is to build a system with multiple interfaces so that it is generic. The trade-off will be that, to get to a sufficiently generic level, you may have to minimize the features you can make available.

In discussing the volatility of internal interfaces and features in COTS components, the BOG members suggested avoiding any dependence on internal product interfaces and functionality in wrappers or customization scripts. If this is not possible, then minimize and isolate these dependencies. The BOG observed that the same solution applies to code developed to work around bugs or deficient capabilities in a COTS product.

The more you use vendor-specific enhancements, the less portable your code becomes and the more you may experience vendor lock problems. Access to unnecessary APIs can also be a problem for future changes, but it can be limited by not exporting the APIs. There is still some risk that developers or users will manage to access these APIs despite your efforts, but project-wide discipline that recognizes this as a problem may help address that.

Wrappers can address error handling to some extent but cannot take the place of a system-wide error handling model. Wrapper code can provide some level of normalization on errors that are returned. Keep in mind that there are limitations on how much information you can derive regarding the source of the errors when using COTS products.

The BOG participants noted that, when handling errors in the context of a protocol that is stateless, there is still state maintained in the various components. This might provide the basis for error handling in this circumstance.

4.4 Transition Phase Issues

Activities during the transition phase address the introduction of the system or product to its ultimate users. The BOG made an important observation that the transition issues listed below have significant impacts on technology, product, and architecture choices. These transition issues must be considered as part of an organization's inception and elaboration activities. It is too late to wait to address these issues late in the life cycle as you deliver the system.

The transition issues for COTS-based systems identified by the BOG are

- *Licensing*

Two major issues must be addressed: (1) what are the licensing arrangements needed to allow your users to use each of the COTS products included in your system, and (2) what are the licensing arrangements and usage implications for any COTS products that are built on top of other COTS products.

For example, a COTS product such as Software Through Pictures (StP) includes a Sybase database. The licensing for StP allows users of a system to use the Sybase database included in StP or to use a Sybase database the end users might already have installed on their operational environment. On the other hand, some systems that include an Oracle database have licensing arrangements that allow their end users to use only the Oracle database bundled into the system.

- *Liability*

The liability concern is best characterized by the following scenario. You build and deliver a system with a COTS component. The COTS component in your system fails during operation and someone dies. Who is liable: you or the COTS vendor? Understanding the consequences of component failure within your system and how you will protect yourself against such occurrences is vital. Resolution options could include selecting an alternate COTS product, custom building the component to meet your stringent requirements, or using an adaptation or wrapper approach that sufficiently insulates the COTS component.

- *Support strategy*

Once a system or product is fielded, who do end users call when they have questions or run into bugs? Understanding the support impact of the technology, product, or architecture choices on the various users' business or mission operations is critical to developing a viable system. The BOG enumerated several scenarios illustrating the support issues that architects and engineers must consider. Does the help desk need to

have an expert on every COTS component that is incorporated into the system? Is all support done in-house, or is some referred to the vendor or to a third party? Can the vendor or third party provide the level and timeliness of support required by your customers?

- *Assumptions of user's installed environment*

How do you deploy a system that needs to determine which products and versions are available? For example, you may want to work with any Web server and with any network browser on the system and with any version of these products. When you are distributing a system based on COTS products, what can you assume that the client already has? This type of situation causes many problems.

There are technical issues in determining what COTS products are installed on a machine and what versions of these products are in use. It would be preferable to do this at runtime or (less preferably) at install time (since a system configuration can change after a system has been installed). For example, Interleaf uses one Oracle database and CASE tools may use a different database version. This can make it impossible to load both tools on a single system at the same time (because of lack of capacity), so the user ends up having to choose between running the different tools. There are management issues as well in terms of what products will be supported.

4.5 Evolution Phase Issues

Activities during the evolution phase address the modifications to your system. COTS-based systems will have new sources of change. One thing that causes such systems to evolve is mission changes or changes in customer requirements. Another is that a new technology becomes available or an older technology becomes obsolete. The evolution issues for COTS-based systems identified by the BOG are

- *Controlling architecture integrity*

Given the potentially high volatility of COTS products and continually emerging technology over the life of a system, how do you ensure that the COTS-based architecture defined initially also evolves without losing its integrity and consistency and degrading into uselessness?

- *Configuration management*

While good configuration management (CM) has long been cited as a necessary element of any successful software-intensive system, many organizations still fail to address their CM needs seriously. The BOG members reported that configuration control is a necessity with COTS-based systems. With the continual evolution of COTS components in a COTS-based system, it is necessary to keep control of the baseline of compatible COTS versions, glue code, and custom components for each release of your system. CM applies not only to the delivered system but also to the development and test environment.

- *Scheduling product upgrades*

How do you schedule upgrades to COTS products? Your system cannot always convert to the latest version as soon as it is available from the vendor. Time and resources are required to understand the characteristics and behavior of a new version and its impact

on your development and test environment and on your delivered system. Then you will have to determine when to allow new versions, to minimize the impact.

- *Engineering technology refresh*

Technology refresh is related to the product upgrade issue although typically focused more broadly. The BOG members discussed changes in technology approach, significant changes in a product that would have correspondingly significant changes in your system, such as a database schema change, and the incorporation of new technology advances. Because of the potential scope of impact, the BOG postulated the need for a "technology refresh team." Such a team might perform the following functions:

- technology watch
- technology and product evaluation
- technical impact analysis

BOG members felt that it is vital for an organization to know what is currently available as well as what is evolving and coming. As an example, how do you take advantage of an emerging technology before the market has had a chance to make products available that implement that technology? For a case in point, consider CORBA. It is not a mature technology, and the availability of CORBA-compatible COTS products is severely limited. This is reflected in the fact that many people are using HTTP because that is where the marketplace currently is. One BOG participant has tried to deal with this situation by dedicating a lot of time to working with advanced or beta releases of products.

- *Migration of legacy code*

Organizations often have many existing systems that they cannot easily replace immediately but that need to be able to take advantage of COTS products. An example cited was the Cheyenne Mountain Upgrade, which needs to become compliant with the Defense Information Infrastructure Common Operating Environment (DII COE). Guidelines are needed on how to take large quantities of legacy code, that may or may not be well engineered, and replace it, possibly in increments, with COTS products. The group noted that legacy code migration places additional requirements and potential compromises on a CBS architecture. This might provide the motivation for identifying an architecture that can be implemented in increments.

4.6 Conclusions of the Engineering and Design BOG

4.6.1 Problem Summaries

In concluding their work, the Engineering and Design BOG summarized the problems that they identified and grouped them into three issue areas: system relationships, component integration, and component visibility.

System Relationship Issues

- How do COTS components relate to
 - requirements
 - systems design and architecture
 - technology
 - integration strategy
- How do you communicate, identify, classify, specify, characterize, and describe a CBS architecture?
- How do you determine the tradeoffs between requirements, architecture, and COTS products?
- How do you evaluate and select COTS components?
- How do you identify the critical, high-risk interrelationships between components?
- How do you handle testing in a COTS based system?

Component Integration Issues

- How do you integrate COTS components into a system to support
 - debugging
 - instrumentation
 - testing
 - upgrade
 - evolution
- How do you determine what are the fragile parts of the architecture?
- How do you select one integration strategy over another?

Component Visibility Issues

- How do you deal with the lack of visibility into COTS components with regard to
 - performance
 - behavior
 - reliability
 - security
 - portability
 - evolvability
 - other non-functional requirements
- How do you discover how components behave or communicate?
- How do you debug systems made up of black boxes?

4.6.2 Observations

One observation of the group was that some of the issues are very general to any type of system and others are quite specific to CBS. Some of the issues have direct COTS-related effects and others are sometimes more widely applicable.

The members of the BOG agreed that using COTS

- does not necessarily simplify the overall design and engineering process
- may actually increase the systems skills required of integrators
- changes the development process
- changes the risk profile associated with system development

They also agreed to these observations:

- The cost savings are in reuse and in evolution.
- Cost increases are in product evaluation and integration.
- Component selection becomes a new critical activity.
- System evolution becomes more complex.
- Prototyping is no longer optional.

5. Acquisition and Management Breakout Group

Members of the Acquisition and Management BOG represented industry, government, and FFRDC perspectives. Appendix A lists the members of the BOG.

Over the course of the two-day workshop, the Acquisition and Management BOG identified eight good practices for acquisition and management personnel implementing CBS approaches. These practices can be summarized as follows:

- Develop flexible central oversight to ensure that commercial components do not undermine the needs of fielded systems.
- Define a full COTS-based life-cycle process.
- Understand and address the differences between government constraints and industry perspectives.
- Make COTS-specific competencies a job requirement for the entire workforce.
- Coordinate with the many groups that are working on COTS issues.
- Establish an information center for COTS issues.
- Develop a set of metrics.
- Explore organizational structures that optimize the use of COTS across multiple projects.

At the outset, the group set its intended scope on the following areas:

- risk management
- acquisition practices
- cost and return on investment (ROI)
- business case(s)
- process(es)

They identified some initial concerns regarding their area of discourse. One was that the areas of acquisition and management are (arguably) less well defined than some of the other areas. For instance, where is acquisition in the five-panel model? (See Appendix C for a description of the five-panel model.) When does it happen? Another concern related to the lack of a practice baseline in this area. That is, if they began their work with a vision of "a CBS practitioner of the future," with what does that contrast today? Finally, they recognized that, for the purposes of the discussions during this workshop, they needed to scope any assertions carefully. For instance, a statement such as "to acquire a COTS-based system, you should ..." will likely need to be revised to include details of domain, size, cost, etc.

This breakout group structured their discussions by first soliciting suggestions for key questions dealing with COTS/component-based systems from each of the participants. The following list of questions resulted:

- What guidance is needed? What are we (i.e., the government and its contractors) missing at the acquisition and management policy level with regard to COTS-based systems?
- What education and understanding are needed for upper management and for the entire workforce? What tools will help furnish this understanding? (And what can one take back from this workshop that might foster such understanding?)
- Where is acquisition in the assembly line? (Where is the acquirer?) (This refers to the graphic of the five-panel model included in this report as Appendix C.)
 - How do the government and contractor negotiate so they can incorporate the new products as they are released? (Look at legal issues, not technical issues.)
 - When does the acquisition process begin? With the initial concept? At Milestone 1?
 - When does it end? Use the actual beginning/ending or formal beginning/ending (as defined in 5000 series)?
 - Is it the same for automated information system (AIS) and weapons systems?
 - How do we (government and contractor) accommodate the (rapid) rate of technology update?
- From the management point of view, how do we measure any dimensions of the COTS acquisition? (We also need to take sustainment into account.)
- From the management point of view, can we gain a deeper understanding of how standards apply and which standards are needed?
 - What do we put on contract to ensure that we get the interfaces we need? Given a legacy system to which COTS items are to be added, interfaces, especially at the F³I (form, fit, function, and interface) level, become very important.
- What guidance can we create or elicit that considers maintenance, update, and post-deployment in early planning? (That is, what can we offer to facilitate the sustainment of these systems? This involves things that we need to do during the front-end acquisition process that influence the whole life cycle. We need to consider things such as licensing, escrow, data rights, management plan, etc.)

The above six sets of questions were grouped into three areas:

- metrics (cost, schedule, etc.) (See Section 5.1.)
- education and training (this became “wisdom and understanding”) (See Section 5.2.)
- guidance and policy (See Section 5.3.)

Each of these three areas was then discussed. As the group proceeded, they noticed that, although they could isolate a small number of pressing problems, the original notion of “solutions to problems,” framed as one of the objectives of the workshop, was not apparent in this subject area. Thus, what appears in the report of this BOG are suggested “pointers toward paths toward potential solutions”; these may not be solutions so much as palliatives.

5.1 Metrics

The BOG agreed that, in order to discuss metrics, they must first establish answers for the following questions, which need to be considered by domain (since the answers may differ depending on the domain of the system):

- Why should I measure?
- What should I measure?
- How do I measure?
- What are the tradeoffs?

5.1.1 Why Should I Measure?

The BOG started by observing that the "I" in the above questions refers to the program manager, and possibly project managers. They then enumerated the following reasons for measuring.

One reason to measure is to gain approval:

- Program managers must measure (cost and schedule) estimates to gain approval.
- Proposal managers must measure (cost and schedule) estimates to gain approval.

It was observed that size may modify the level of detail of the above measurements, although another participant commented that size should *not* modify the level of detail, and some discussion ensued. In general, this discussion was an indication of the increased rigor that is needed for all acquisitions.

Another reason to measure is to manage the project and support tradeoffs:

- Project/program managers must measure (cost, schedule, etc.) to manage the project. One participant commented that we must also measure the quality of the process; we need to do quality assurance way up front on products and services.
- Project/program managers must measure (cost, schedule, etc.) to evaluate or justify tradeoffs.

Other reasons to measure include

- Managers must measure (cost, schedule, etc.) to preserve historical data, and to improve the process that is being used.
- Project/program managers must measure whether the system or component does what it should.
- Managers must measure whether or not the project is performing as expected.

5.1.2 What Should I Measure?

Armed with this understanding of “why” (and who), the BOG next explored the question, “What should I measure?” in more detail. While Table 1 does not give all-inclusive lists, it does convey much of the group’s conversation on this point.

The last set of measures in the table (regarding process) led one participant to raise a question: What is the new life cycle? Another asked whether we know what processes have been used. Examples of different processes seem to fall into three categories:

1. new start with a COTS-based system
2. migration from a legacy system to a new architecture and a COTS base
3. selective upgrade (e.g., component ‘X’ is bad, replace it [initial or sustainment])

What is measured here may be different depending on the process. One participant added that earned value should still apply, but that one needs to know what the process is and what the tasks are before earned value can be effectively applied. The method for measuring is the same, but the tasking will be different. Fit analysis or gap analysis can be used; one of the participants reported doing this recently.

5.1.3 How Do I Measure?

Assuming the above provided some understanding of “what,” the BOG next explored the question “How should I measure?” in more detail. Again, the following list is not meant to be all inclusive.

Methods (strategies) and mechanisms for measuring that the group discussed included

- fit analysis (a mechanism)
- paying a little to buy more information
- counting defects (e.g., define quality attributes in terms of the customer perspective)
- counting function points

One participant reported on an experience at the US Air Force’s Electronic Systems Command (ESC) that is an example of paying a little to buy information. Initially ESC did not have a good idea of the cost to do a particular project. So they wrote a vision statement, then gave that to five contractors, with a small amount of money, for a period of three months, with the instructions to go out and build a prototype. Each of the five contractors had expertise in a different technology. None of those five contractors was guaranteed participation in the next round. At the end of three months, the resulting prototypes from the five contractors were opened to the public. ESC used the knowledge from this to do the real contract RFP.

Costs	licenses purchase price cost of operation cost of development ² cost of deployment integration dependency cost ³ bureaucratic costs ⁴ market research/market survey cost training testing
Quality	product quality service quality system quality performance documentation, training (e.g., number of help desk calls, average response time) quality of company (vendor or organization providing product or service) ⁵ product stability vendor stability standards conformance (How well does product conform to standards?)
Schedule	plans vs. actuals activities and sequence paths progress ⁶
Process	productivity of integrators productivity of managers productivity of testers

Table 1: Some Things to Measure

² To help with calculating this, one might do fit analysis to see how much you need to develop.

³ Dependency cost is the cost of all the other items on which the product might depend and other implications (licenses, etc.).

⁴ Bureaucratic costs are the actual acquisition costs, especially in concept exploration, etc.

⁵ Possible measures include the viability of the company, vendor, or organization providing the product or service, their market share (if appropriate), their CMM level, etc.

⁶ One potential method for measuring this is earned value.

It was noted there was a need to identify some means to measure everything on the list of what to measure (see Table 1), although the BOG did not have the time to be that thorough. It was also noted that a survey could be used as a mechanism to gain measurement information.

5.1.4 What Are the Tradeoffs?

With the above coverage of “how”, the BOG finally explored the question, “What are the tradeoffs?” in more detail. This too is not meant to be all inclusive.

Tradeoffs enumerated by the group included

- tight integration vs. flexibility (lock yourself into one [tightly integrated] or go with many). Considerations include
 - number of vendors
 - maintenance implications
 - legal issues

There is a cost of flexibility; in addition, one is also assuming that using multiple products makes a system truly flexible.

- general contractor vs. do-it-yourself
- prioritized and rated requirements
- go with the flow (of what is available or prominent in the marketplace) vs. use of military standards
- cost and schedule vs. desired requirements (“desirements”) (It is important to involve users/functionals here.)
- schedule vs. functionality

5.2 Wisdom and Understanding

This area was initially labeled “education and training.” In the ensuing discussion, other phrases describing this area included workforce development, workforce migration and skills development, and developmental issues related to technology infusion and management awareness. In terms of educating upper management, the basic question is how to better inform them; how will enlightenment occur? Peter M. Senge’s book, *The Fifth Discipline: The Art and Practice of the Learning Organization* [Senge 90], was recommended to the group.

The following captures the BOG’s thoughts in this area. The ideas listed are further annotated with *needed* versus *clarify*, to distinguish those things that do not exist (and thus are “needed”) from those that exist but are not clearly understood.

- lessons learned database (*needed*)
 - needs to be across programs
 - needs to be synthesized and abstracted (plusses and minuses of the lessons learned)
- all levels of peer testimonials (*needed*)
- better descriptive rhetoric (to better capture reality)(*needed*)
- a CBS approach includes both benefits and risks (*clarify*)
- internalization of lessons learned and wisdom (*needed*)
- incentives for the above internalization (Is it money? If not, then what?) (*needed*)
- removal of disincentives (e.g., data rights, legal issues) (*needed*)
- process for technology infusion (*needed*)
 - method for fielding new system as it comes out; how does one infuse new technology projects?
 - in both a micro (infusing the use of COTS products into a specific project) and a macro sense (infusing the use of COTS into the entire DoD acquisition process)
- information gathered across organizations (synthesis) (*needed*)
 - example of partnership: Marines 3-D distributed training system program sharing the cost of development with a commercial company
 - Navy putting together an acquisition center of excellence
- implication that the “C” (commercial) makes a real difference in COTS (*clarify*)
- good education, from a number of perspectives (*needed*)
 - requires both ongoing education and a recognition that ongoing education is a cost of doing business
 - education in COTS product (vendor type), as well as in COTS usage
 - in terms of senior management awareness (“This is not as simple as it sounds”); consider sessions at Software Technology Conference—at plenary session where leaders talk about COTS issue; do a better job orchestrating this; target opportunities
 - just-in-time learning
 - best way to learn is from people you respect

The group then discussed the question of whether there should be information clearinghouses to help disseminate the acquired information and knowledge. It was noted that there are a number of hard tradeoffs that decision makers must face (e.g., one contractor/tightly integrated versus many contractors/flexible). What means exist to disseminate the information that we find? Mark Schaeffer's area (the Systems Engineering Offices in DoD Acquisition and Technology) may be a good place to learn more about the effort required to insert more software material—including information on COTS issues—into Defense Systems Management College (DSMC) courses.

5.3 Policy and Guidance

Three broad areas where guidance is needed were articulated by the group:

- contracting
- standards
- sustainment (life-cycle strategy, planning, and definition)

5.3.1 Guidance for Contracting

There are many areas regarding contracting where misunderstanding exists (and hearsay sometime prevails). The BOG looked at contracting in the context of the Information Technology Management Reform Act (ITMRA) (now known as the Clinger/Cohen Act) and the constraints imposed by it. They tried to answer the questions of what guidance for contracting exists and what is still needed.

The discussion resulted in the following list of things for which guidance is needed:

- upgrades
- data rights (which involves FAR rewrites)
- use of escrow accounts
- licensing
 - Has it been licensed by person's name, by number of seats, by enterprise, etc.? Know your usage; you can negotiate, and make sure you have a clause in the agreement so you can renegotiate.
 - What does the license agreement include? Does it include servicing, all upgrades, incremental upgrades, etc.? What occurs if an upgrade is skipped?
 - Are there license monitors? What occurs if the license is not renewed (promptly)? People may not know about licensing and options they may have (e.g., enterprise licensing).
- performance specifications, performance-based specifications
 - direction to make maximal use of commercial products versus direction to do performance specification (Note: a performance specification tells "what" not "how"; i.e., it is not doing design specification.)
 - What are the impacts of having COTS products furnished as a solution to the performance specification? what are the implications in the sustainment phase?
- liability
- vendor alliances

Necessary actions revealed by the discussion include

- Synthesize and disseminate guidance if it exists.

- Create guidance if it is lacking.
- Dispel hearsay!

5.3.2 Guidance Regarding Standards

The discussion regarding standards resulted in a set of questions that need to be answered:

- What is the relationship between DII/COE and
 - the various COTS-related policies
 - various COTS products
- What is the relationship between DII/COE and COTS-based applications?
 - What is the vendor's goal with respect to COTS-based systems? The required minimum level of compliance with DII/COE is Level 5, but its creator, the Defense Information Systems Agency (DISA), indicates a need to have, as a goal, Level 8. (BOG discussion ensued as to whether or not one had to have source code to do segmenting. One opinion was one should be able to do a single segment with COTS [without the need for source code]. Lockheed Martin is preparing a tool to do segmentation without having source.)
- How do COTS products and a COTS approach relate to DISA's idea of interoperability?
- How do open systems and an open systems approach relate to COTS and a COTS approach?
- How do you evaluate for standards compliance?
 - This issue relates back to technical and institutional goals of why you want this system in the first place; also goes back to how you write the requirements.
 - Evaluating the compliance with standards has a strong tie-in to technology and product evaluations.

5.3.3 Guidance Regarding Sustainment

One clear conclusion of the BOG was that we must expand the management focus to include the update part of the life cycle, not merely the initial purchase:

- Focus on the update part of the life cycle, not only the front end of the cycle.
 - When do you update, how do you plan for replacement, etc., in budgeting?

If you support a hundred systems worldwide, and need to deal with different versions of software, just trying to keep up with operating system releases could be a challenge. As an example, the Army skipped one release of a product and it is estimated it took them three times as long to get back in sync.

Another comment was that you should not skip a release, unless you never intend to upgrade. The vendor may have a tool that allows you to skip upgrades, although you

might have to pay all the upgrade fees (but you would not have to actually do all the upgrades).

Look at the domain, as well as the life cycle of the product.

The May 1997 issue of CrossTALK (Volume 10, Number 5) has an article about Portable Reusable Integrated Software Modules (PRISM), *C++ Component Integration Obstacles* by Bruce D. Swanson and John G. McManus. It provides information about how they found a path around problems associated with upgrades.

- What are the tradeoffs and what are the pitfalls?
- How independent are you (can you be) from the vendor's release cycle?
- Be ready for the "domino" effect of different vendors' upgrades.
- With respect to releases, there are three different strategies, each of which has a technical side and a business side:
 1. one-time buy ("user" takes over maintenance) One participant commented: "If the COTS piece is a user piece, forget it."
 2. take every release (vendor cycle is your cycle)
 3. take selective releases (many dependencies)

The choice among these three strategies can be affected by many considerations:

- A. the level of user interface
- B. the domain
- C. the interface with other programs
- D. the number and distribution of users
- E. security needs

You can look at the characteristics of your programs and suggest which of the above three COTS release strategies to adopt. It might be useful to develop a matrix of 1-2-3 versus A-B-C-D-E, as in Figure 3.

The members of the group provided several examples of things that can be done regarding sustainment.

- One group bought life-cycle support for the product; they negotiated this up front and paid one time.
- Another issue is installation. If you have tens or hundreds of thousands of users and you have a six-month release cycle, you cannot use Strategy 2 (take every release) above. Use of Strategy 2 may work, though, if the release cycle is 18 months.
- An example from Air Force Material Command (AFMC) described how the users bought their own releases instead of waiting for AFMC to give it to them, and that caused problems.
- Another member commented that if you use configuration management to distribute software automatically, the user does not have a choice as to when upgrade occurs; e.g., a user logs in and automatically gets the latest release. However, this approach has its own set of problems.

	Level of user interface	Domain	Interface with other programs	Number, distribution of users	Security needs
One-time buy					
Every release					
Selective releases					

Figure 3. Strategy Analysis Matrix

5.4 “Answers”

In preparation for the report to the plenary session later in the day, the initial breakout group posed to themselves the question, “What can we do about this?” With a caveat to “get real” on what they proposed (i.e., is it something that has a reasonable chance or that we can influence?), they labeled the resulting thoughts as “answers.”

- Central oversight (with flexibility) is needed to ensure that the use of COTS does not undermine the needs of fielded systems (especially with regard to interoperability).
 - The group explored why COTS was such a hot issue now. In terms of “faster, better, cheaper,” one participant noted that the overriding concern appeared to be (in this order) “cheaper,” next “faster,” and finally, “better.”
 - Another participant noted that COTS and standards appeared to be mutually exclusive.
 - Another noted that part of the interoperability issue will be solved by more standard products and that DII/COE is part of the solution.
- A full COTS-based life-cycle process for COTS-based systems, including requirements gathering, testing, and evaluation, must be defined (as in DOD-STD- 2167A).
 - We need to be clear on testing issues initially and in the sustainment portion of the full life cycle.
- Recognition of the differences between government constraints and the industry perspective must be part of any acquisition and management strategy. This implies that the processes in the bullet above may have significant differences from those used by industry. (It was noted this was not so much an answer as a wise statement.)
- COTS-specific core competencies must be a job requirement for the entire workforce dealing with software (program managers, etc.).

- What skill sets are required of the (entire) workforce?
- Facilitate coordination between the many groups that are working "the COTS issue."
- Develop a characterization matrix for program/attribute factors.
- Establish an information center for COTS-related issues.
- Work toward a better way to describe COTS-related issues.
 - It was noted that work towards this goal can be part of the life-cycle process, and it also ties to the core competencies item.
- Develop a set of useful and useable metrics that relate to COTS-based systems.
- Explore the viability of organizational structures that optimize the use of COTS products and technologies across multiple projects (programs).
 - We should not only optimize the initial use of COTS products, but also their sustainment.
 - How do we get an organization out of a stovepipe mentality and get support for infrastructure? (It was noted that the requirements approval process is a stovepipe, and the budget approval process is a stovepipe, but they are *different* stovepipes.)

5.5 Conclusions of the Acquisition and Management BOG

The BOG concluded with actions that they thought were feasible to take—in some cases, perhaps by members of the BOG themselves. As can be seen, many of them are taken from parts of the discussions reported on in the preceding sections.

- AI1. Look at the Buying Commercial and Nondevelopmental Items (CANDI) manual and propose software-related items for inclusion.
- AI2. Explore the fit between DII/COE and COTS. (This was in the context of guidance regarding standards.)
- AI3. Develop a characterization matrix for program/attribute factors.
- AI4. Establish an information center for COTS-related issues.
- AI5. Establish coordination between the many groups that are working on "the COTS issue."
- AI6. Work toward a better way to describe COTS-related issues.

6. Workshop Conclusion

We are thoroughly convinced that this workshop was useful and productive. The participants that were brought together all contributed significantly to the growing body of CBS knowledge. In some cases they suggested new hypotheses; in others, they served to confirm the hypotheses of others. The ideas that took greater shape during these two days will form the basis for future work investigating COTS-based systems. Future workshops in this series will depend on the results of this work to advance the maturity of CBS practices for the whole community.

The SEI would like to thank all of the participants in this workshop. Without their contributions no progress would be possible. We are looking forward to a continued association with all of these valuable members of the CBS community.

Appendix A: Workshop Participants

A.1 Technology and Product Evaluation Breakout Group

Participants in the Technology and Product Evaluation BOG included

1. Scott Bachman, Department of Defense
2. John C. Dean, National Research Council of Canada
3. Anthony Earl, Software Engineering Institute (**scribe**)
4. Peter H. Feiler, Software Engineering Institute
5. Dave Gluch, Software Engineering Institute
6. David Kyle, Joint Engineering Design Management Information and Control System (JEDMICS)
7. Tom Loggia, Lockheed Martin
8. Ed Morris, Software Engineering Institute (**facilitator**)
9. Gerri Regazzi, Tessada and Assoc., Inc.
10. Steve Roberts, Planning Research Corporation (PRC)
11. Louis Rose, Software Productivity Consortium (SPC)
12. Tom Timberlake, Boeing
13. Kurt Wallnau, Software Engineering Institute (**co-facilitator**)
14. Marvin Zelkowitz, National Institute of Standards and Technology (NIST)/University of Maryland

A.2 Engineering and Design Breakout Group

Participants in the Engineering and Design BOG included

1. Lisa Brownsword, Software Engineering Institute (**facilitator**)
2. Gary Chastek, Software Engineering Institute
3. Gail Cochrane, TRW
4. Mike Gagliardi, Software Engineering Institute
5. Robert Monroe, Carnegie Mellon University
6. David Remkes, PRC
7. Caroline Ruso, PRC
8. Robert Seacord, Software Engineering Institute (**scribe**)
9. Mark Tappan, SPC
10. Mark Vigder, National Research Council of Canada

A.3 Acquisition and Management Breakout Group

Participants in the Acquisition and Management BOG included

1. Linda D. Brown, OASD (C3I)/IT
2. David J. Carney, Software Engineering Institute (**facilitator**)
3. Mark Cavanaugh, Software Engineering Institute
4. Judith A. Clapp, The MITRE Corporation
5. Eileen C. Forrester, Software Engineering Institute
6. Steve Nelson, JEDMICS Program Office
7. Patricia A. Oberndorf, Software Engineering Institute
8. Carol A. Sledge, Software Engineering Institute (**scribe**)
9. Nancy Solderitsch, Lockheed Martin C2 Integration Systems
10. LCDR Doug Swanson, Naval Reserve Information Systems Office

Appendix B: Unified Modeling Language

The Unified Modeling Language (UML) is a language for specifying, constructing, visualizing, and documenting the artifacts of a software-intensive system.

First and foremost, the Unified Modeling Language fuses the concepts of Booch, OMT, and object-oriented software engineering (OOSE). The result is a single, common, and widely usable modeling language for users of these and other methods.

Second, the Unified Modeling Language pushes the envelope of what can be done with existing methods. In particular, the UML authors targeted the modeling of concurrent, distributed systems, meaning that UML contains elements that address these domains.

Third, the Unified Modeling Language focuses on a standard modeling language, not a standard process. Although the UML must be applied in the context of a process, it is the experience of its creators that different organizations and problem domains require different processes. (For example, the development process for shrink-wrapped software is an interesting one, but building shrink-wrapped software is vastly different from building hard-real-time avionics systems upon which lives depend.) Therefore, the efforts concentrated first on a common metamodel (which unifies semantics), and second on a common notation (which provides a human rendering of these semantics). The UML authors will not necessarily standardize a process, although they will continue to promote a development process that is use-case driven, architecture centric, iterative, and incremental.

Appendix C: The Five-Panel Model

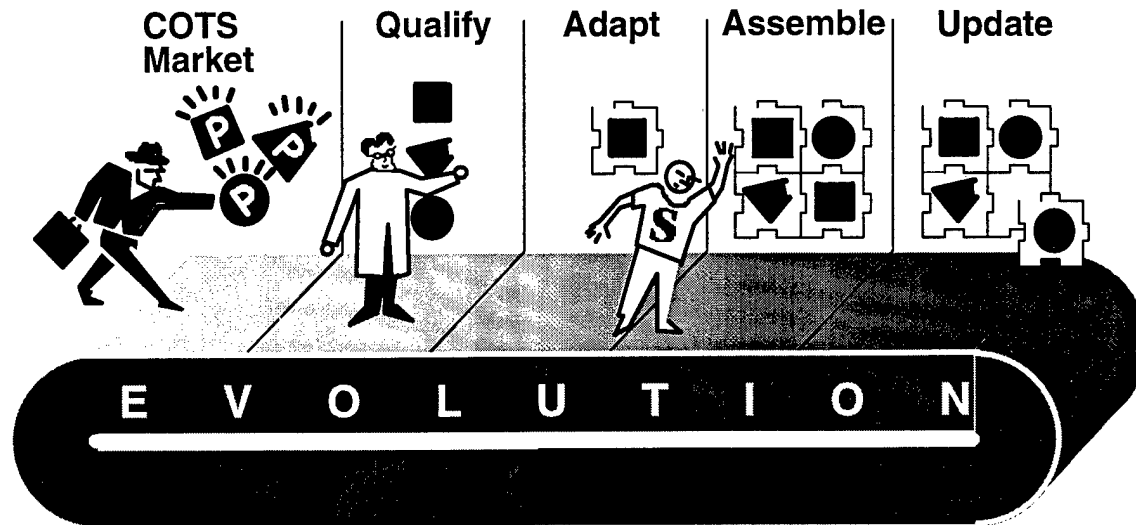


Figure 4: The Five-Panel Model

The model pictured in Figure 4 is called the five-panel model. The five panels represent various activities that address different aspects of turning a set of COTS components into a COTS-based system.

- The COTS Market panel deals with the market survey and analysis activities that determine what are the viable candidates for a particular component, from both a business and a technical perspective.
- The Qualify panel activities investigate the hidden interfaces and other characteristics and features of the candidate products. The result of this discovery process is to reveal the necessary information to make a selection and identify possible sources of conflict and overlap, so that the component can be effectively assembled and evolved.
- The Adapt panel activities amend the selected components to address potential sources of conflict. The figure implies a kind of component “wrapping,” but other approaches are possible (e.g., mediators and translators).
- The Assemble panel shows the integration of the adapted components into an architectural infrastructure. This infrastructure will support component assembly and coordination, and differentiates architectural assembly from ad hoc “glue.”
- The Update panel acknowledges that new versions of components will replace older versions; in some cases, components may be replaced by different components with similar behavior and interfaces. These replacement activities may require that wrappers be rewritten, and they suggest the advantage of well-defined component interfaces that reduce the extensive testing otherwise needed to ensure that the operation of unchanged components is not adversely affected.

Appendix D: List of Acronyms

ADL	Architecture description language
AFMC	Air Force Materiel Command
API	Application program interface
BOG	Breakout group
CASE	Computer-aided software engineering
CBS	COTS-based system
CM	Configuration management
CMM	Capability Maturity Model
CORBA	Common Object Request Broker Architecture
COTS	Commercial off-the-shelf
DCOM	Distributed Component Object Model
DII/COE	Defense Information Infrastructure/Common Operating Environment
DISA	Defense Information Systems Agency
DMSO	Defense Modeling and Simulation Office
DSMC	Defense Systems Management College
ESC	Electronic Systems Command
FAR	Federal Acquisition Regulations
FFRDC	Federally funded research and development center
HLA	High-level architecture
HTTP	Hypertext transfer protocol
ITMRA	Information Technology Management Reform Act
ODBC	Open Data Base Connectivity
OMG	Object Management Group
OMT	Object Modeling Technique

OOSE	Object-oriented system engineering
PRISM	Portable Reusable Integrated Software Modules
RFI	Request for information
RFP	Request for proposal
RFQ	Request for quote
ROI	Return on investment
SAAM	Software Architecture Assessment Method
SEI	Software Engineering Institute
StP	Software Through Pictures
TRAC2ES	TRANSCOM Regulating and Command and Control Evacuation System
UML	Unified modeling language

References

- [Abowd 97] Abowd, G., et al. *Recommended Best Industrial Practice for Software Architecture Evaluation* (CMU/SEI-96-TR-025, ADA 320 768). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, January 1997.
- [Garlan 95] Garlan, D. et al. "Architectural Mismatch (Why It's Hard to Build Systems Out of Existing Parts)," 170-185. *Proceedings, 17th International Conference on Software Engineering*. Seattle, WA, April 23-30, 1995. New York: Association for Computing Machinery, 1995.
- [Kazman 96] Kazman, Rick; Abowd, Gregory; Bass, Len; and Clements, Paul. "Scenario-Based Analysis of Software Architecture," *IEEE Software* 13, 6 (November 1996): 47-55.
- [Kruchten 95] Kruchten, P. "The 4+1 View Model of Architecture." *IEEE Software* 12, 6 (November 1995): 42-50.
- [Kruchten 96] Kruchten, P. *A Rational Development Process* [online]. Available Worldwide Web <<http://www.rational.com>>.
- [Lichota 97] Lichota, R.; Vesprini, R.; and Swanson, B. "PRISM Product Examination Process for Component Based Development." *Proceedings of the Fifth International Symposium on Assessment of Software Tools and Technologies*. Pittsburgh, PA, June 2 - 5, 1997. Los Alamitos, CA : IEEE Computer Society.
- [Senge 90] Senge, Peter. *The Fifth Discipline: The Art and Practice of the Learning Organization*. New York: Doubleday/Currency, 1990
- [Rational 97] Rational Software Corporation. *Unified Modeling Language: UML Summary* [online]. Available Worldwide Web <<http://www.rational.com/uml>>.
- [Shaw 95] Shaw, M. "Making Choices: A Comparison of Styles for Software Architecture." *IEEE Software* 12,6 (November 1995): 27-41.
- [Shaw 96] Shaw, Mary. "Truth vs. Knowledge: The Difference Between What a Component Does and What We Know It Does." *Proceedings of the 8th International Workshop on Software Specification and Design*. March 1996.
- [Wright 97] Wright. [online]. Available Worldwide Web <<http://www.cs.cmu.edu/afs/cs/project/able/www/wright/index.html>>.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (LEAVE BLANK)		2. REPORT DATE November 1997	3. REPORT TYPE AND DATES COVERED Final
4. TITLE AND SUBTITLE Workshop on COTS-Based Systems		5. FUNDING NUMBERS C — F19628-95-C-0003	
6. AUTHOR(S) Patricia Oberndorf, Lisa Brownsword, Ed Morris, Carol Sledge			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213		8. PERFORMING ORGANIZATION REPORT NUMBER CMU/SEI-97-SR-019	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) HQ ESC/AXS 5 Eglin Street Hanscom AFB, MA 01731-2116		10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES			
12.A DISTRIBUTION/AVAILABILITY STATEMENT Unclassified/Unlimited, DTIC, NTIS		12.B DISTRIBUTION CODE	
13. ABSTRACT (MAXIMUM 200 WORDS) This report documents the proceedings of the first Workshop on COTS-Based Systems, held at the Software Engineering Institute (SEI) June 10-11, 1997. It describes the workshop activities, the discussions of three breakout groups, and some general conclusions reached by participants in the workshop.			
14. SUBJECT TERMS acquisition, COTS (commercial off-the-shelf), COTS architecture,		15. NUMBER OF PAGES 60	
COTS-based systems, COTS system design, COTS system management, evaluation		16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL